**vue开发不可避免走过的坑汇总**

**1、不要在循环中使用v-if：尽管这看起来很直观，但它会导致一个巨大的性能问题——VueJS优先考虑 v-for 而不是 v-if 指令。**

**我们应该在遍历模板中的数据之前对其进行过滤。有两种非常相似的方法：**



* **使用计算属性**

<ul>

<li v-for='products in productsUnderFifty' :key='product.\_id' > {{ product.name }}</li>

</ul>

<script>

export default {

data () {

return {

products: []

}

},

computed: {

productsUnderFifty: function () {

return this.products.filter(product => product.price < 50)

}

}

}

</script>

* **使用过滤方法**

<ul>

<li v-for='products in productsUnderPrice(50)' :key='product.\_id' > {{ product.name }} </li>

</ul>

<script>

export default {

data () {

return {

products: []

}

},

methods: {

productsUnderPrice (price) {

return this.products.filter(product => product.price < price)

}

}

}

</script>

**2、生成随机组合**

function randomString(len) {

len = len || 32;

var $chars = 'ABCDEFGHJKMNPQRSTWXYZabcdefhijkmnprstwxyz2345678'; /\*\*\*\*默认去掉了容易混淆的字符oOLl,9gq,Vv,Uu,I1\*\*\*\*/

var maxPos = $chars.length;

var pwd = '';

for(i = 0; i < len; i++) {

pwd += $chars.charAt(Math.floor(Math.random() \* maxPos));

}

return pwd;

}

**3、懒加载**

<script>

$(function() {

var pageNum = 1,

canAdd = true;

$(".department-box").scroll(function(event) {

var scrollTop = event.currentTarget.scrollTop;

var offsetHei = $(this).height();

var tablesHei = $(this).find("table").eq(0).height();

var pageSize = 50;

if((scrollTop + offsetHei + 200) > tablesHei && canAdd) {

pageNum++;

canAdd = false;

//根据listType 确定appId 用appId还是appCode 0是工作站,用appCode

var appId = "";

var listType = "1";

if(listType == "1") {

appId = "SMARTZHlcsp4nzcW9MY";

} else {

appId = "";

}

var pfId = "228659633701650432";

$.ajax({

url: "/zciid-deliverable/business/businesscaasmanagement/deliverablesLibrary-data",

//?appId=SMARTZHlcsp4nzcW9MY&pfId=228659633701650432

data: {

appId: appId,

pfId: pfId,

listType: listType,

pageNum: pageNum,

pageSize: pageSize

},

type: "POST",

dataType: "html",

beforeSend: function() {

JLayer.loading();

},

success: function(data) {

if(data != '') {

var trLength = $(".department-box table tbody").eq(0).find("tr").length;

$(".department-box table tbody").eq(0).append(data);

$("tr").eq(0).find("td").each(function(index, item) {

var isShow = $(item).is(":visible");

if(isShow) {

$(".department-box table tbody").eq(0).find("tr").each(function(ind, ite) {

if(ind >= trLength) {

$(ite).find("td").eq(index).show();

}

})

} else {

$(".department-box table tbody").eq(0).find("tr").each(function(ind, ite) {

if(ind >= trLength) {

$(ite).find("td").eq(index).hide();

}

})

}

})

canAdd = true;

JLayer.closeLoading();

} else {

pageNum--;

JLayer.closeLoading();

setTimeout(function() {

canAdd = true;

}, 2000)

}

},

error: function(data) {

console.log(data.message);

JLayer.closeLoading();

}

})

}

})

})

</script>

**4、如果你有一个表格组件，你可以使用这个功能如下：**

<template>

...

<my-table>

<template #row={ item }>

/\* 一些内容，你可以在这里自由使用“item” \*/

</template>

</my-table>

...

</template>

**5、$on(‘hook:’) 可以帮助你简化代码**

删除事件监听器是一种常见的最佳实践，因为它有助于避免内存泄露并防止事件冲突。

如果你想在 created 或 mounted 的钩子中定义自定义事件监听器或第三方插件，并且需要在 beforeDestroy 钩子中删除它以避免引起任何内存泄漏，那么这是一个很好的特性。下面是一个典型的设置：

mounted () {

window.addEventListener('resize', this.resizeHandler);

},

beforeDestroy () {

window.removeEventListener('resize', this.resizeHandler);

}

使用 $on('hook:')方法，你可以仅使用一种生命周期方法（而不是两种）来定义/删除事件。

mounted () {

window.addEventListener('resize', this.resizeHandler);

this.$on("hook:beforeDestroy", () => {

window.removeEventListener('resize', this.resizeHandler);

})

}

**6、你应该始终验证你的Prop**

验证 Props 是 Vue 中的基本做法之一。

你可能已经知道可以将props验证为原始类型，例如字符串，数字甚至对象。你也可以使用自定义验证器——例如，如果你想验证一个字符串列表：

props: {

status: {

type: String,

required: true,

validator: function (value) {

return [

'syncing',

'synced',

'version-conflict',

'error'

].indexOf(value) !== -1

}

}

}

**7、动态指令参数**

Vue 2.6的最酷功能之一是可以将指令参数动态传递给组件。假设你有一个按钮组件，并且在某些情况下想监听单击事件，而在其他情况下想监听双击事件。这就是这些指令派上用场的地方

**8、重用相同路由的组件**

开发人员经常遇到的情况是，多个路由解析为同一个Vue组件。问题是，Vue出于性能原因，默认情况下共享组件将不会重新渲染，如果你尝试在使用相同组件的路由之间进行切换，则不会发生任何变化。

const routes = [

{

path: "/a",

component: MyComponent

},

{

path: "/b",

component: MyComponent

},

];

如果你仍然希望重新渲染这些组件，则可以通过在 router-view 组件中提供 :key 属性来实现。

<template>

<router-view :key="$route.path"></router-view>

</template>

**9、把所有Props传到子组件**

可让你将所有 props 从父组件传递到子组件。如果你有另一个组件的包装组件，这将特别方便。所以，与其把所有的 props 一个一个传下去，你可以利用这个，把所有的 props 一次传下去：

代替：

<template>

<childComponent v-bind="$props" />

</template>

代替：

<template>

<childComponent :prop1="prop1" :prop2="prop2" :prop="prop3" :prop4="prop4" ... />

</template>

**10、把所有事件监听传到子组件**

如果子组件不在父组件的根目录下，则可以将所有事件侦听器从父组件传递到子组件

如果子组件位于其父组件的根目录，则默认情况下它将获得这些组件，因此不需要使用这个小技巧。

<template>

<div>

...

<childComponentv-on="$listeners" />...

<div>

</template>